May 1993
Vol. 4 No. 5

COBB
INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

DOS marches on: Version 6 arrives

By Van Wolverton

Despite recurring rumors to the contrary, DOS refuses to succumb to the onslaught of its glitzy pretenders and continues to roll out new features, gaining speed and improving its manners all the while. Windows and OS/2 might dress flashier and cause more talk, but Version 6 shows that there's still a lot of life left in the Disk Operating System for the IBM PC and compatibles that first ran on a customer's machine nearly 12 years ago.

DOS 6 offers something for everyone. Power users get three new commands and startup options that help find problems in CONFIG.SYS and AUTOEXEC.BAT. Tinkerers get some nifty utilities that used to require other programs. Windows users get their own versions of three utility programs. Novices get automatic memory management and a really useful online help facility. In fact, we all benefit from each new feature. DOS is improving with age.

Three new commands

Each new command fills an obvious need. In fact, *Inside DOS* has published several articles over the years showing how to create batch files or programs that achieve the effect of all three new commands.

A handy batch file tool

The first new command is CHOICE, which waits for the user to press a key, then converts the keystroke to an ERRORLEVEL that can be checked with the IF command. You can use the CHOICE command when you want to write an interactive batch file. It seems that more workarounds have been developed to make up for the lack of this command than for any other reason. We expect that the CHOICE command will probably appeal to a lot of DOS users.

Easy moving and renaming

MOVE is just what its name implies—a command that moves a file from one disk or directory to another, deleting the original. In a nice touch, MOVE also lets you rename a directory, something we've never been able to do before from the command prompt.

A powerful deleting option

The last new command, DELTREE, deletes an entire branch of the directory tree, whether it's a single directory or one with many subdirectories and files. Its power is a welcome addition to the kit of DOS tools, but use the command with care—it offers only one confirming prompt.

Four new utilities

DOS is known for its spare look—a bare screen except for the C:\ prompt. The DOS 5 Editor, Shell, and QBasic weakened the stereotype a bit; DOS 6 further erodes it by adding four utility programs that sport their own mouse interface, complete with menus and dialog boxes. By licensing these new utilities from Central Point Software and Symantec, Microsoft continues a trend started in Version 5 when it licensed UNDELETE from Central Point Software.

A better backup

MSBACKUP replaces the BACKUP command. Now you can define a backup procedure with a minimum of hassle, specifying the files and directories you want to back

IN THIS ISSUE

up by choosing them from a list with the mouse. You can then carry out the procedure as frequently as you like with even less effort. It's just the sort of encouragement we need to back up more often.

Virus protection

MSAV (Microsoft Anti-Virus) scans all the files on the disk you specify for known viruses. Central Point Software plans to make periodic updates to the virus database available so you can keep your protection current.

A disk defragmentor

DEFRAG rearranges the way files are stored on a hard disk to pack them into a single area. This ordering means that DOS doesn't have to work with a lot of small chunks of files interspersed with available disk space (which is known as a *fragmented* disk) and can handle disk operations more quickly.

Doubling disk space

DoubleSpace installs a large (51 Kb) device driver to compress and expand files on the fly so that your hard disk can hold up to twice as much data as it now can. If you don't have a commercially available compression program, such as Stacker from Stac Electronics, and you're about to buy a new hard disk or a new system because you've run out of disk space, this feature alone could justify the cost of upgrading to DOS 6.

More and better help

Online help has been greatly expanded. Instead of simply displaying a screenful of dense text describing the syntax of the command and explaining its parameters, online help now starts a separate program that displays several screens of more lucid descriptions. You'll find tips on how to use a command and examples of common uses. Better still, hypertext-like links let you jump quickly to related topics. If you type *help* at the DOS prompt without adding the name of a command, Help displays an index of all help topics. You can jump directly to any topic just by clicking on it with a mouse or by highlighting it and pressing [Enter].

Optimizing your configuration

DOS 6 introduces a new utility that will help you optimize your system's memory. MemMaker's mission is simple—to make best use of the memory available in your system—and its options are blessedly few. Fire it up, answer some questions, and a few minutes later (after restarting your system a couple of times), MemMaker has changed CONFIG.SYS and AUTOEXEC.BAT to put as many programs in high memory as possible. This automated system is definitely an improvement over the handwork that was necessary with Version 5 to accomplish the same thing.

Almost as helpful are the options now available each time you start the system. Want to start without process-

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices	Domestic .... $49/yr. ($6.00 each) Outside U.S. .... $69/yr. ($8.50 each)
Phone	Toll free .... (800) 223-8720 Local .... (502) 491-1900 Customer Relations Fax .... (502) 491-8050 Editorial Department Fax .... (502) 491-4200
Address	You may address tips, special requests, and other correspondence to The Editor, *Inside DOS* 9420 Bunsen Parkway, Suite 300 Louisville, KY 40220 For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to Customer Relations 9420 Bunsen Parkway, Suite 300 Louisville, KY 40220
Postmaster	Second class postage is paid in Louisville, KY. Send address changes to *Inside DOS* P.O. Box 35160 Louisville, KY 40232
Back Issues	To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the U.S. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all the articles that have appeared in *Inside DOS*.

Staff

Editor-in-Chief	Suzanne Thornberry
Contributing Editors	Van Wolverton David Reid
Editing	Elizabeth Welch Timothy E. Hampton
Production Artist	Julie Jefferson
Design	Karl Feige
Publications Manager	Tara Dickerson
Circulation Manager	Brent Shean
Publications Director	Linda Baughman
Editorial Director	Jeff Yocom
Publisher	Douglas Cobb

ing either CONFIG.SYS or AUTOEXEC.BAT? Just press [F5] when DOS displays its new message *Starting DOS*. How about a chance to decide whether to carry out the commands in CONFIG.SYS one by one? Just press [F8] at startup, and DOS prompts you for each command, then asks whether to carry out AUTOEXEC.BAT.

The icing on the cake is the ability to define different sets of configuration commands in CONFIG.SYS and choose which set to carry out each time you start or restart the system. Here's another capability so badly needed that several different workarounds have been devised, including some commercially available products.

Windows and portable users gain, too

DOS 6 even lets Windows users share the wealth. When you install DOS, you're given the option of installing Windows versions (either in addition to or in place of the DOS versions) of the undelete, anti-virus, and backup utility programs. Pay some attention when you make this decision, though, because the Windows versions take up 1.9 Mb of disk space and the DOS versions take up 1.3 Mb. In fact, installing both versions of all three utilities takes up about 3 Mb of hard disk space, while all the rest of DOS is another 4 Mb or so. You may not want to nearly double the amount of disk space DOS requires just to get both versions of these three utilities.

If you install DOS on a portable computer, you'll find two features of interest. First, a device driver named POWER.EXE tries to conserve battery power as much as possible. If your computer conforms to the Advanced Power Management (APM) specification, Microsoft claims as much as a 25-percent savings; nonconforming computers can expect much less, perhaps as little as a 5-percent savings.

The other feature for portable users is a program named INTERLNK that makes possible high-speed file transfer using the computer's existing ports. DOS doesn't include the necessary cables, so programs like Laplink—which include the necessary cables—offer a more complete answer. However, if you've got the cables (or can get them), you can transfer files between two PCs by using only DOS 6.

It isn't a perfect solution

There are some confusing inconsistencies in the user interface of the DOS add-ons. The Shell, DOS Editor, QBasic, and DoubleSpace are quite similar, but MSAV and UNDELETE (licensed from Central Point Software) have a slightly different look and feel; MSBACKUP and DEFRAG (licensed from Symantec) differ in more ways. Then there's the new online help, which doesn't look or feel much like any of them. It can be a little bit confusing to shift from one interface to another.

But the price is right—all the new commands and utilities are included with DOS 6, which will probably cost less than $50 (to upgrade) for the first few months, maybe longer. Considering that you could easily spend $200 or more for the various utilities bundled with DOS 6, the new operating system looks like the biggest bargain of the year.

Keep in mind, however, that the utility programs have fewer features and often don't equal the performance of their commercially available counterparts. Individual utility programs and packages of utilities are available from many software companies. Each package offers its own strengths and weaknesses, but almost all of them can justifiably claim to offer better features and performance than one or more of the utilities bundled with DOS.

Even though DOS 6 doesn't offer the same compelling reasons to upgrade that DOS 5 did—not to mention the fact that hardly any of us are struggling to cope with the unmourned Version 4—upgrading is still the right choice. The new features of Version 6 make DOS a bit easier to use every time you turn your system on, and your DOS shop will gain a host of shiny new tools. DOS 6 is a good investment. ■

Special characters help filenames sort to the top

If you use the DOS Shell, Microsoft Windows, or another shell program that displays lists of files in directories, you may find that it's sometimes inconvenient to scroll through a long list looking for the file you need. In most cases, the shell program will display files in alphabetical order. Of course, the files you consider important won't necessarily be at the top of the list—unless your important files happen to begin with the letter *a*.

Here's a trick for making your most used files appear at the top of the directory listing: Rename the files so that the filename begins with a special character, such as !, @, #, $, or &. (Of course, you can't use the ? or * wildcards in the new filename.) The next time you list the files in the directory, the one beginning with the special character will appear at the top of the list.

For example, suppose you often update a file named C:\DOCS\ZOO.DOC. Renaming the file will prevent you from having to scroll to the end of the C:\DOCS directory listing when you want to open the file through the DOS Shell. You can quickly rename the file in the DOS Shell by selecting the filename and issuing the Rename command from the File menu ([Alt]F,N). Then, simply type the new name, *!zoo.doc*, in the New Name text box and press [Enter]. When you do, the DOS Shell will move the !ZOO.DOC file to the top of the C:\DOCS directory listing.

Creating a flexible batch file for choosing screen colors

Have you ever wanted an easy way to change screen colors, without having to remember—or look up—arcane ANSI codes? As we showed in the article "Using the PROMPT Command to Color Your DOS Screen," which appeared in the May 1992 *Inside DOS*, one way to change colors more easily is to create a batch file that sets the screen colors to your favorite combination. However, the COLOR.BAT file we showed you in that article contains only *one* color combination. For example, you could set COLOR.BAT to display white text on a blue background.

Although COLOR.BAT provides an easy way to color your screen when you boot up or exit an application, the batch file isn't flexible. If you'd like to use another color combination for variety or for emphasis, you'll need to look up the new ANSI codes and issue another long PROMPT command. To overcome this limitation, we've developed the COLORS.BAT batch file shown in Figure A. In this article, we'll show you how COLORS.BAT works and how to use it.

Getting ready

Of course, you'll need a color monitor in order to take advantage of the COLORS.BAT file. You'll also need to install the ANSI.SYS driver in your CONFIG.SYS file, if you haven't already done so. The ANSI.SYS driver should be in your C:\DOS directory, so you can install the driver by placing in your CONFIG.SYS file the line

```
device=c:\dos\ansi.sys
```

If you're able to load drivers into upper memory, you can use the DEVICEHIGH statement instead of DEVICE, as in

```
devicehigh=c:\dos\ansi.sys
```

Using the DEVICEHIGH statement will prevent the ANSI.SYS driver from taking up as much conventional memory as it would if you used a DEVICE statement.

COLORS.BAT basics

COLORS.BAT allows you to enter a natural sounding command to set your screen colors. When you run the batch file, you type *colors*, followed by the name of the color you want to use for the text. Next, you type *on*, followed by the name of the color you want to use as the background color. For example, to choose white text on a blue background, you'd type

```
C:\>colors white on blue
```

and then press [Enter]. COLORS.BAT will set the screen colors and clear the screen. Note that you enter three parameters with the COLORS command.

In our example, white (the foreground color) corresponds to the first parameter (%1) and blue (the background color) corresponds to the third parameter (%3). We included a second parameter, the word *on*, to help new users remember to enter the text color before the background color.

In a nutshell, the COLORS.BAT file works by checking to see whether you entered a valid color as the foreground or background color. If you entered a valid color, COLORS.BAT will enter the corresponding ANSI color code for you with a PROMPT command. Now, let's look more closely at each of the commands that make up the COLORS.BAT file.

A closer look

As with most batch files, COLORS.BAT begins with a command to turn ECHO off, preventing DOS from displaying each command of the batch file. The REM statement that follows is simply a brief note reminding you what the batch file does.

The FOR statement

```
for %%a in (ON on On) do if "%%a"=="%2" goto :OK
```

checks to see if the word *on* is the second parameter you enter when you run COLORS.BAT. Basically, this FOR statement is just checking to see if you use the correct syntax when you run COLORS.BAT. (For more information on using FOR commands, see the article "Using FOR Instead of IF Can Make Your Batch Files More Flexible," which appears on page 8.) If you typed *ON*, *on*, or *On*, the FOR statement will jump to the :OK label.

However, if you didn't type the word *on*, the FOR statement will carry out the instructions in the :HELP section. The CLS command will clear the screen, and a series of ECHO commands will display the help message

```
To set your screen colors, type "colors," the text color
you want, "on," and then the background color you want.
For example, to choose black text on a red background, enter:

                    colors black on red


You can choose from the following colors:

          Black          Red
          Green          Yellow
          Blue           Magenta
          Cyan           White
```

```
If you'd like a brighter version of your text color,
enter HIGH after the colors command. For example:

            colors yellow on blue high
```

The help message simply reminds users how to tell COLORS.BAT which colors to set. If you use the DOS Editor to create COLORS.BAT, you can insert either tabs or spaces to indent lines within the messages. In this section, the batch file issues the ECHO command followed immediately by a period to display a blank line onscreen. After the ECHO commands display the message, the command

```
goto :END
```

quits the batch file and deletes any color variable.

If you enter the batch file with the correct syntax—that is, if you enter the word *on* as the second parameter—the batch file will jump to the :OK section. This section tests to see which colors you want to use and then sets the colors. The section consists mainly of FOR statements. The first eight of these FOR statements check to see which color you entered as the first parameter. When a FOR statement matches a color to the parameter you entered, it sets up an environment variable to store the ANSI code for that color. For example, let's suppose you run the COLORS.BAT file by entering

```
C:\>colors blue on cyan
```

When you run COLORS.BAT with this statement, *blue* is %1 (the first parameter), *on* is %2, and *cyan* is %3. When the batch file processes the fifth of the FOR statements in the :OK section

```
for %%a in (BLUE blue Blue) do if "%%a"=="%1"
    set text=34
```

it will set the TEXT environment variable to 34, which is the ANSI code for blue text. If you look closely at the FOR statement, you can see that COLORS.BAT sets the variable when it processes the second value for %%a from the set *(BLUE blue Blue)*, because *blue* matches the first parameter (%1) you entered when running the batch file.

The ninth through 16th FOR statements check to see which color you entered for the background (represented by the third parameter). Returning to our example of *colors blue*

Figure A

```
@echo off
rem COLORS.BAT lets you choose colors for your DOS screen.
for %%a in (ON on On) do if "%%a"=="%2" goto :OK
:HELP
cls
echo.
echo.
echo To set your screen colors, type "colors," the text color
echo you want, "on," and then the background color you want.
echo For example, to choose black text on a red background, enter:
echo.
echo                    colors black on red
echo.
echo.
echo   You can choose from the following colors:
echo.
echo              Black            Red
echo              Green            Yellow
echo              Blue             Magenta
echo              Cyan             White
echo.
echo.
echo  If you'd like a brighter version of your text color,
echo  enter HIGH after the colors command. For example:
echo.
echo                   colors yellow on blue high
echo.
echo.
goto :END
:OK
for %%a in (BLACK black Black) do if "%%a"=="%1" set text=30
for %%a in (RED red Red) do if "%%a"=="%1" set text=31
for %%a in (GREEN green Green) do if "%%a"=="%1" set text=32
for %%a in (YELLOW yellow Yellow) do if "%%a"=="%1" set text=33
for %%a in (BLUE blue Blue) do if "%%a"=="%1" set text=34
for %%a in (MAGENTA magenta Magenta) do if "%%a"=="%1" set text=35
for %%a in (CYAN cyan Cyan) do if "%%a"=="%1" set text=36
for %%a in (WHITE white White) do if "%%a"=="%1" set text=37
for %%a in (BLACK black Black) do if "%%a"=="%3" set back=40
for %%a in (RED red Red) do if "%%a"=="%3" set back=41
for %%a in (GREEN green Green) do if "%%a"=="%3" set back=42
for %%a in (YELLOW yellow Yellow) do if "%%a"=="%3" set back=43
for %%a in (BLUE blue Blue) do if "%%a"=="%3" set back=44
for %%a in (MAGENTA magenta Magenta) do if "%%a"=="%3" set back=45
for %%a in (CYAN cyan Cyan) do if "%%a"=="%3" set back=46
for %%a in (WHITE white White) do if "%%a"=="%3" set back=47
for %%a in (HIGH high High) do if "%%a"=="%4" goto :CONTRAST
if not exist %text% do goto :HELP
if not exist %back% do goto :HELP
@echo on
prompt $e[0;%text%;%back%m
goto :CLEANUP
:CONTRAST
@echo on
prompt $e[1;%text%;%back%m
:CLEANUP
prompt $p$g
@echo off
cls
:END
set text=
set back=
```

COLORS.BAT sets environment variables to the codes for the colors you choose.

on cyan, the batch file finds that %3—the third parameter—is *cyan*. So, the statement

```
for %%a in (CYAN cyan Cyan) do if "%%a"=="%3" set back=46
```

sets the BACK environment variable to 46, the ANSI code for a cyan background.

The last FOR statement

```
for %%a in (HIGH high High) do if "%%a"=="%4"
    goto :CONTRAST
```

checks to see if you entered a fourth parameter. As you may recall from looking at the :HELP section, you can enter the word *high* after the colors in order to make the text brighter. Returning to our example, if you entered *colors blue on cyan high*, this FOR statement will detect the fourth parameter you typed and then jump to the :CONTRAST section.

After the FOR statements, COLORS.BAT checks to see if the TEXT and BACK environment variables exist by executing the statements

```
if not exist %text% do goto :HELP
if not exist %back% do goto :HELP
```

If COLORS.BAT wasn't able to create one of the variables, it will display the messages in the :HELP section and quit. You'd see this message if you entered a color name that wasn't valid. For example, you'd see the message if you entered

```
C:\>colors purple on cyan
```

since *purple* isn't a valid color name.

If you didn't type *high* as the fourth parameter, COLORS.BAT will carry out the remaining instructions in the :OK section:

```
@echo on
prompt $e[0;%text%;%back%m
goto :CLEANUP
```

The first command turns ECHO on. As you may recall, ECHO must be turned on so that DOS can receive the ANSI command for setting screen colors. The PROMPT command then sends the ANSI escape sequence for setting the colors. In this command, *$e* (which represents the Escape character) and the left bracket introduce the ANSI codes. The *0* at the beginning of the string turns off any special attributes for the text, including the high-intensity attribute. After a semicolon, the environment variable TEXT sets the code for the text color. Another semicolon separates the text color from the background color, which is contained in the BACK environment variable. At the end of the PROMPT command, *m* tells ANSI.SYS that the codes you've entered control the monitor. After carrying out the prompt command, the batch file will jump to the :CLEANUP label.

Before we look at the :CLEANUP section, however, let's examine the :CONTRAST section:

```
:CONTRAST
@echo on
prompt $e[1;%text%;%back%m
```

As we mentioned earlier, COLORS.BAT will carry out the commands in the :CONTRAST section if you type the word *high* at the end of your COLORS command. The first command in the section simply turns ECHO on so that DOS can receive the ANSI commands that control the monitor. Then, the :CLEANUP section provides brighter text by substituting *1* for *0* at the beginning of the PROMPT command that sets your screen colors.

Whether or not you've chosen higher contrast, COLORS.BAT will carry out the instructions in the

Creating COLORS.BAT with the DOS Editor

Although you can create COLORS.BAT with any text editor that allows you to save ASCII-only text, the DOS Editor's Cut and Paste commands will make it easier for you to create the batch file.

After you've typed the first FOR statement, which tests to see if you chose the color black, you can select the line by clicking on it and dragging the mouse. If you don't have a mouse, you can place your cursor on the *f* at the beginning of the line, then hold down the [Shift] key and press ↓. Once you've selected the line, you can press [Ctrl][Insert] to copy the line. (Or, if you prefer, you can issue the Copy command from the Edit menu.) Then, move the cursor to the next line and press [Shift][Insert] to paste a copy of the line below the original. (Or issue the Paste command from the Edit menu.) Now, you can edit the copied line. To do so, change all occurrences of *black* to *red* and change the number at the end of the line to *31*. You can continue editing the FOR statements in this manner through the eighth one. Then, you can copy the first eight and paste them after the line that tests for white. You can edit the lines in this second section of FOR statements by changing all occurrences of *%1* to *%3* and by adding 10 to the number at the end of the line. For example, you change the number *31* to *41* in the line that checks for red.

When you've typed the remaining lines of the batch file, you'll need to save it on a directory that's on your path. For example, if you keep batch files in your C:\BATCH directory, save the batch file as C:\BATCH\COLORS.BAT.

:CLEANUP section (shown below) after executing the PROMPT command:

```
:CLEANUP
prompt $p$g
@echo off
cls
:END
set text=
set back=
```

The first command in the :CLEANUP section restores your normal system prompt. (We've assumed that you normally use the *pg* prompt, which displays the current path, followed by a greater than sign. Of course, you can substitute your favorite system prompt for this command.) Next, the :CLEANUP section turns ECHO off and clears the screen.

The :END label precedes the commands that set the TEXT and BACK environment variables equal to nothing. Deleting the variables in this way helps conserve your environment space. We placed the :END label *before* these SET commands so that the :HELP section could also delete any stray environment variables that might result if you make a typo when you try to run COLORS.BAT.

Using COLORS.BAT

We've shown you the basics of using COLORS.BAT. You simply enter *colors*, the name of the text color you want, the word *on*, and the name of the background color you want. You can add the word *high* to the end of the command if you'd like a brighter text color, which can provide more contrast for some color combinations. So, if you'd like bright yellow text on a blue background, you'd enter the command

```
C:\>colors yellow on blue high
```

Running COLORS.BAT when you boot up

You can use COLORS.BAT from the DOS prompt whenever you'd like to change your screen colors. But you can also call COLORS.BAT to change the screen colors while you run another batch file. For example, you'll probably want to call COLORS.BAT from your AUTOEXEC.BAT file to color the screen as your computer boots. Returning to our example, you could set bright yellow text on a blue background by placing the following command in your AUTOEXEC.BAT file:

```
call colors yellow on blue high
```

Be sure to include the CALL command before the batch file name. The CALL command ensures that DOS returns to the AUTOEXEC.BAT file after running COLORS.BAT to change your screen colors.

Running COLORS.BAT from other batch files

You can also use COLORS.BAT to change colors within your own batch files. For example, suppose you want to add color to LOG.BAT. (Van Wolverton showed you how to create LOG.BAT in the article "LOG.BAT Helps You Keep Track of Important Events," which appeared in last month's issue.) Let's also suppose you want to display cyan text on a blue screen when you make entries in the log. To do so, you substitute the following command for the CLS command that appears under the :OK label:

```
:OK
call colors cyan on blue
```

At the end of the :OK section, you insert another command restoring the colors you normally use. For example, to restore the screen colors to bright yellow on blue, you'd place the following command above the :END label in LOG.BAT:

```
call colors yellow on blue high
:END
```

If you'd like to use COLORS.BAT to set off a message, you'll need to include a PAUSE command before you restore the colors. Otherwise, COLORS.BAT will clear the message from the screen before you get a chance to read it. Returning to our LOG.BAT example, you can change the colors to bright white text on a red background by adding the commands shown in red to the message section of LOG.BAT:

```
if not "%1"=="" goto :OK
call colors white on red high
echo.
echo You must enter the name of a file as a
echo parameter (for example, LOG FONE.TXT).
echo If the file doesn't exist, it is created.
echo.
pause
call colors yellow on blue high
goto :END
```

As you can see, you place the first CALL command after the IF statement, which jumps to the :OK label if you entered a parameter (%1) when you ran LOG.BAT. But, if you don't enter a parameter, LOG.BAT will clear the screen to a red background, then display the help message in bright white text. The PAUSE command will display the message *Press any key to continue...* at the end of the echoed lines.

Conclusion

In this article, we've shown you how to create a batch file that allows you to change screen colors easily without entering ANSI codes. We've also shown you how to use COLORS.BAT from other batch files. ■

Using FOR instead of IF can make your batch files more flexible

In the article "Creating a Flexible Batch File for Choosing Screen Colors," which begins on page 4, we use FOR statements instead of IF statements to detect the colors you want COLORS.BAT to set for your screen. You might wonder why we didn't test for colors by using an IF statement like

```
if "%1"=="black" set text=30
```

Unfortunately, this IF statement isn't case sensitive. So, if you ran COLORS.BAT with the command

```
C:\>COLORS BLACK ON RED
```

the IF statement wouldn't consider the first parameter you entered—*BLACK*—equal to *black*.

We wanted COLORS.BAT to detect the color names whether users type them in all lowercase, all uppercase, or capitalized. To do so using only IF statements, you'd need three different statements for each color:

```
if "%1"=="BLACK" set text=30
if "%1"=="black" set text=30
if "%1"=="Black" set text=30
```

A more efficient alternative is to use a FOR command. As you may know, the FOR command takes the form

```
for %%a in (set) do [your commands here]
```

In COLORS.BAT, we simply substitute the three forms of the color name in the set *(BLACK black Black)*. Then, after the DO command, we use an IF statement to set the TEXT variable if one of the colors in the set equals the first parameter, as in

```
for %%a in (BLACK black Black) do if "%%a"=="%1"
    set text=30
```

This FOR statement allows you to enter *BLACK*, *black*, or *Black* as the first parameter when you run the COLORS.BAT file.

You can apply these techniques in your own batch files. Just replace IF statements with FOR statements when you'd like your batch files to recognize uppercase, lowercase, or capitalized parameters.

DOS BASICS

Comparing files with the FC command

Suppose you have two versions of the same file—which is the one you need? Usually, you'll be able to figure out which file you want: You just check the date on the directory listing and keep the most recent file. But sometimes, you might want to know exactly which lines differ between two versions of a file. Fortunately, DOS' FC command lets you quickly compare the contents of two files.

In this article, we'll show the basics of using the FC command for comparing files. We'll show some options the FC command provides for displaying the differences between files. We'll also show you a special case in which you'll want to override FC's default comparison.

FC basics

Comparing two files with the FC command can be as simple as typing *fc*, followed by the names of the two files you want to compare. (You'll also need to include the path with the filename when the files are in different directories.) For example, suppose you want to compare two documents, NESSIE.DOC and NESSIE1.DOC, which are both in the C:\DOCS directory. From the C:\DOCS directory, issue the command

```
C:\DOCS>fc nessie.doc nessie1.doc
```

When you press [Enter], the FC command will display the lines that differ between the files, as well as one line before and one line after the differences. For example, the FC command may display the following when it compares NESSIE.DOC and NESSIE1.DOC:

```
Comparing files NESSIE.DOC and NESSIE1.DOC
***** NESSIE.DOC
dreamed of: Tracking Elvis to his secret hideaway in Loch
Ness, Scotland. Elvis' well-known references to the
"Hound Dog" may actually have been veiled references to
"Nessie," as we at the Prying Eye affectionately know the
creature
***** NESSIE1.DOC
dreamed of: Tracking Elvis to his secret hideaway in Loch
Ness, Scotland. As Bridgeway Getajob has noted, Elvis'
references to the "Hound Dog" may actually have been
```

```
veiled references to "Nessie," as we at the Prying Eye
call the creature.
*****
```

As you can see, the FC command displays all the lines in the two documents that differ plus a line above and below the changed text. (By the way, the FC command includes changes in line breaks in the differences it reports.)

A couple of helpful switches

If you don't want to see the entire section of changed text, you can add the /A switch after *fc* to tell the FC command to display only the first and last lines in each set of differences. You might also want to include the /N switch, which adds line numbers to the report. For example, here's what the FC command would display if you use the /A and /N switches when you compare NESSIE.DOC and NESSIE1.DOC:

```
C:\DOCS>fc /a /n nessie.doc nessie1.doc
Comparing files NESSIE.DOC and NESSIE1.DOC
***** NESSIE.DOC
  23:  dreamed of: Tracking Elvis to his secret hideaway
in Loch
...
  27:  creature
***** NESSIE1.DOC
  23:  dreamed of: Tracking Elvis to his secret hideaway
in Loch
...
  27:  creature.
*****
```

When you use the /A switch, the FC command displays the lines that match in the files and uses an ellipsis (...) to represent the lines that don't match. The line numbers show you that the differences occur between lines 23 and 27 in the files.

A special case: The CONFIG.SYS file

One feature of the FC command is that it "decides" for you whether to compare files as ASCII text or as binary code. In most cases, FC will assume that you want to compare files as ASCII text. However, if you compare files with an EXE, COM, SYS, OBJ, LIB, or BIN extension, FC will default to a binary comparison. (As you know, the EXE and COM extensions indicate executable programs and command files, while the SYS extension indicates a device driver. Programming languages create and use binary files with the OBJ, LIB, and BIN extensions.)

You'd need to change from the default comparison when you compare two versions of your CONFIG.SYS file. Suppose you've installed a new software package that alters your CONFIG.SYS and saves your previous CONFIG.SYS as CONFIG.BAK. When you reboot, you notice that another program isn't working properly. You could look through your CONFIG.SYS file to find references to the new program, but it's even easier to let the FC command show you the differences. However, if you issue the command *fc config.sys config.bak*, you'll see a long string of hexadecimal numbers but no text. This binary information won't help you find out what's wrong with your CONFIG.SYS file. Fortunately, you can add the /L switch to tell FC to treat the CONFIG.SYS file as a text file rather than a binary file. When you add the /L switch to the command to compare the files, you might see something like

```
C:\>fc /l config.sys config.bak
Comparing files CONFIG.SYS and CONFIG.BAK
*****CONFIG.SYS
dos=high,umb
device=c:\cdrom\mscdex.exe
device=c:\ansi.sys
*****CONFIG.BAK
dos=high,umb
device=c:\dos\ansi.sys
```

In this case, the FC command shows you that the new CONFIG.SYS file has added a device statement installing the MSCDEX driver. Debugging this statement should help you get your system working properly again.

Conclusion

In this article, we've introduced you to the FC command, a useful utility included with DOS. We've shown you the basics of comparing files and how to use the switches that make the FC command more versatile.

Why not COMP?

You may wonder why we're focusing on using the FC command instead of the COMP command for comparing files. In general, the FC command is more flexible than the COMP command. Here's a brief rundown on COMP's limitations:

- COMP lets you compare only those files that are the same size.
- COMP displays only ten differences before quitting.
- COMP doesn't offer as many options for displaying differences.

CHKDSK /F can cause problems with large hard disk drives

An early version of MS-DOS 5.0 contained two commands—CHKDSK and UNDELETE—that *may* be incompatible with some hard disk drives larger than 100 Mb. The problems with the commands will affect you only if

- your CHKDSK and UNDELETE files are dated before 11-11-91
- your hard disk uses a 128-Kb file allocation table (FAT)

Even if both conditions are true, the CHKDSK command causes problems *only* if you use it with the /F switch, which tells DOS to convert lost clusters of data to files. Under these conditions, the CHKDSK command may lose some of the data contained on your hard disk. The early version of the UNDELETE command may work unreliably on large hard disks.

At the end of this article, we'll show you how to contact Microsoft if you need updated files. But first, we'll show you how to determine whether the bug will affect your system.

Checking your system

You can find the date your CHKDSK command was created by changing to your DOS directory and issuing the command *dir chk**. DOS will display the file's directory listing, which will look something like

```
CHKDSK   EXE      16200    04-09-91 5:00a
```

In this listing, the date *04-09-91* indicates that the CHKDSK file is from the first release of MS-DOS 5.0. Thus, you may experience problems when you use the CHKDSK command with the /F switch. If you see the listing

```
CHKDSK   EXE      16200    11-11-91 5:00a
```

your CHKDSK command is safe to use, no matter what size hard disk drive your system uses.

The UNDELETE command on your system should have the same date as the CHKDSK command. If you'd like to make sure, you can issue the command *dir undel** from the DOS directory.

Is your hard disk compatible?

Even if your CHKDSK and UNDELETE commands originated before November 1991, you may not have anything to worry about. The early versions of CHKDSK and UNDELETE in MS-DOS 5.0 function fine with most hard disks. You'll need to replace the files only if you have a logical drive that falls into one of the size ranges shown in Table A. (A *logical drive* is your whole hard disk if you have only a C: drive or any of the other drives—such as the D: or E: drives—if you've partitioned your hard disk. You don't need to check RAM disks.)

Table A

127-129 Mb
254-258 Mb
508-516 Mb
1,018-1,030 Mb
2,035-2,061 Mb

If a logical drive on your hard disk falls into one of these size ranges, the drive may be incompatible with two DOS 5 commands.

If you aren't sure whether your system could be affected, you can check the size of your logical drives by entering the FDISK command followed by the /STATUS switch:

```
C:\>fdisk /status
```

The /STATUS switch tells the FDISK command to display a report that looks something like

```
DiskDrv      Mbytes   Free     Usage
1            114      0        100%
      C:     84
      D:     30
```

If any number in the *Mbytes* column falls within the ranges shown in Table A, you should request the updated files from Microsoft. If not, your CHKDSK and UNDELETE commands will work properly on your system, no matter what date the commands originated.

If you need the new files . . .

If you've discovered that your system might be affected by the bug, you can request updated CHKDSK and UNDELETE files from Microsoft. One way to get the files is to call Microsoft's Consumer Sales department at (800) 426-9400. Be sure to have the serial number from your MS-DOS installation diskettes handy when you call, and let the representative know whether you need 3.5" or 5.25" diskettes.

Getting the updated files is even easier if you have a modem. You can obtain the files through the Microsoft Download Service. You can connect to the service by calling (206) 936-6735 at up to 9,600 bps. There's no additional connect fee, though you'll need to pay long-distance charges. Be sure to select 8 data bits, no parity, and 1 stop bit for your communications package when you call. If you've never used the MSDL service, you'll be prompted to complete a brief registration form before you can access the files. Then, you'll need to select the Windows and MS-DOS section, where you can download the PD0646.EXE file, which contains the updated CHKDSK and UNDELETE files and a note on using the files.

Once you've downloaded your files or received them on a diskette, you simply copy the new versions of the commands over the old versions. For example, if the new files are on a diskette in your A: drive, you'd issue the commands

```
C:\>copy a:\chkdsk.exe c:\dos
C:\>copy a:\undelete.exe c:\dos
```

If you use a COPY macro that runs SAFECOPY.BAT instead of the COPY command (as we explained in the article "Guarding Against the Hidden DOS Trap," which appeared in the February 1993 issue), simply type a space before COPY to override the macro.

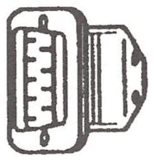
PC BASICS

Serial versus parallel ports

If you've ever installed a new peripheral on your PC, you've probably come across the terms *serial* and *parallel*. These terms describe the type of connectors you can use to let your PC communicate with your new equipment. When you install a new piece of equipment, you'll connect a cable from the peripheral to a serial or parallel port in your PC. Let's briefly look at some differences between the two types of connectors.

You can usually recognize a serial connector as a 9-pin plug in the back of your PC. (PCs built before 1984 may have 25-pin serial connectors.) Serial connectors, also known as RS-232C connectors, are often used to attach a mouse or modem to PCs. Figure A shows a diagram of a typical serial connector.

Figure A

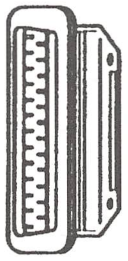


Serial connectors usually contain nine pins.

Each serial port is named *COM* and is followed by a number. For example, if your PC has two serial ports, they'll be named *COM1* and *COM2*. You might need to know which COM port a device is attached to when you install software that uses the device. For example, suppose your modem is attached to COM2 and you want to install a new communications package. You'll need to make sure the communications software looks for the modem at the COM2 port.

A parallel connector, on the other hand, is bigger, and you can recognize a parallel connector by its 25-pin plug, as shown in Figure B. Parallel cables contain more wires than serial cables do, and they also transmit data much more quickly.

Figure B



Parallel connectors contain 25 pins.

Most people attach their printers to parallel ports to take advantage of the faster data rate. In fact, parallel ports are so strongly associated with printers that the ports are named with *LPT*, an old abbreviation for *line printer*. Your first parallel port will be named *LPT1*; if you have a second port, it's named *LPT2*.

While we're on the subject of connectors, we'd like to caution you to always keep track of the cable connected to your monitor and note exactly which plug it connects to. Plugging your monitor into a standard serial port—rather than its special video connector—can damage your equipment.

LETTERS

A quirk of the DOS clock

I have discovered a problem with the DOS clock that seems to be inherent to DOS. The clock is updated or initialized each time the system boots up by "reading" the system clock. However, if I leave the system on for several days, the DOS clock seems to lose five to ten seconds over a week. If I leave the system running long enough, even the date can get out of sync. Do you have an explanation or solution short of rebooting every day?

Robert M. Miller
Odenton, Maryland

Observant DOS users like Mr. Miller often wonder if the variance in the DOS clock is caused by a malfunction in their system or if it's just a limitation of DOS. The short answer is that the DOS clock just isn't a very accurate timepiece. If you'd like to know more, let's look at some background on the clocks your system uses.

If you have a 286-based PC or a more recent model, your computer depends on its CMOS clock to keep track of the time when you turn off your PC. The CMOS clock can do this because it's powered by a battery. When you boot up, your DOS clock reads the time CMOS reports and then begins keeping time by counting the 18.2 "ticks" generated every second by your system's circuitry. Based on this data, the DOS clock reports a time that DOS can use for many functions, such as timestamping files.

The bottom line is that the DOS clock can routinely lose up to 20 seconds a day. Certain applications can make the DOS clock even more inaccurate. Leaving your computer unused for a day can have an even more surprising effect: DOS may not increment the day, even though the time will be just a few seconds behind. When you leave the computer alone, DOS doesn't call the CMOS-based system clock, and DOS loses track of the day.

As Mr. Miller reports, rebooting your computer causes the DOS clock to resynchronize itself with the CMOS clock, resetting the clock to the correct date and time. Rebooting is the only way to correct the problem if you rely solely on DOS. However, if you leave your computer running Windows, your system should keep track of the day.

If you find the timekeeping problem very inconvenient, you might want to check into some third-party clock drivers. You install these clock drivers through your CONFIG.SYS file. Some are available through computer bulletin boards, including the IBM forum on CompuServe.

Another clock utility is available from a company named Pacific Standard Time. After you install their CLOCK.EXE utility, you run it a few times a week, resetting the time if necessary. The CLOCK utility "learns" how your system's clock varies from the correct time. Each day, you can correct the time by entering the command *clock run*. For more information on the utility, you can call Pacific Standard Time at (408) 246-0589.

Some CMOS setup utilities include NUM LOCK option

In the February 1993 issue, you published the article "Creating Commands for Controlling NUM LOCK and CAPS LOCK." Although the commands you discussed can control NUM LOCK and CAPS LOCK, I've discovered a simpler way to control whether NUM LOCK is turned on or off when my computer boots.

First, I access my computer's setup program by pressing [Delete] while the system boots. Then, I choose Advanced CMOS Setup and Alter Options to Make System Work. On my system, the option for controlling NUM LOCK is called System Boot Up Num Lock. If I select Yes, the system will turn on NUM LOCK; if I select No, the system will turn off NUM LOCK. After I change the option to No, I press [Escape] and choose Write to CMOS and Exit. The Setup program asks me to confirm my choice, so I press Y and [Enter] to select Yes. When you've changed the CMOS, the system will reboot, and the changes will be in effect.

Linda E. Sullivan
Phoenix, Arizona

Thanks for the tip! As Ms. Sullivan notes, some newer computers (mostly 486 and later models) allow you to set NUM LOCK through their CMOS setup program. As we mentioned in "Preparing for Battery Failure Will Help You Get Back to Work Quickly," which appeared in the February 1993 issue of *Inside DOS*, you might access your CMOS settings by simply pressing [Ctrl][Alt][Enter], [Ctrl][Alt][Escape], or some other key combination.

If you decide to explore changing your NUM LOCK setting through the CMOS setup, avoid changing any other information—especially the setting for your system's memory and disk drives. If you accidentally type incorrect information in these fields, your computer may not function properly. ■

Printed in the USA
Printed on recycled and recyclable paper
50% recycled fiber including 10% post-consumer waste paper